Linköpings Tekniska Högskola
Institutionen för Datavetenskap
Filip Strömbäck och Magnus Nielsen

10 januari 2020

TDDI16
Datastrukturer och algoritmer
Datortentamen (DAT1)
2020-01-10, 08–12

Examinator:	Erik Nilsson
Jour:	Filip Strömbäck (telefon 013-28 27 52)
Antal uppgifter:	6
Max poäng:	40 poäng
Preliminära gränser:	Betyg 5 = 35p, 4 = 27p, 3 = 20p.
Hjälpmedel:	Inga hjälpmedel tillåtna!

VÄNLIGEN IAKTTAG FÖLJANDE

- Observera att betygsgränserna kan komma att justeras, i samtliga kurser.
- Vid frågor om tidskomplexitet, svara alltid på den form som är mest relevant.
- Du får själv välja om du vill skriva din lösning på papper eller på dator.
- Lösningar till olika problem skall placeras enkelsidigt på separata blad, eller i egen fil. Skriv inte två lösningar på samma papper eller i samma fil. Delproblem får dela papper / fil.
- Om inte annat framgår ska indexering av arrayer / listor börja från 0.
- Papper: Sortera lösningarna innan de lämnas in.
- Filer: Skicka in som lösning till rätt problem i tentaklienten, och döp filen till ett passande namn (exempelvis uppg1.txt).
- **MOTIVERA DINA SVAR ORDENTLIGT:** avsaknad av, eller otillräckliga, förklaringar resulterar i poängavdrag. **Även felaktiga svar kan ge poäng** om de är korrekt motiverade.
- Om ett problem medger flera olika lösningar, t.ex. algoritmer med olika tidskomplexitet, ger endast optimala lösningar maximalt antal poäng.
- Papper: Lämna plats för kommentarer.
- SE TILL ATT DINA LÖSNINGAR/SVAR ÄR LÄSLIGA. Oläsliga lösningar beaktas ej.

Lycka till!

1. **Hashtabeller** (5 p)

Vi har en hashtabell med linjär adressering och några element instoppade. Hashfunktionen är h(x) = x mod size (arrayindex står under arrayen för tydlighets skull).

Null	11	21	3	2	25	5	27	26	Null
0	1	2	3	4	5	6	7	8	9

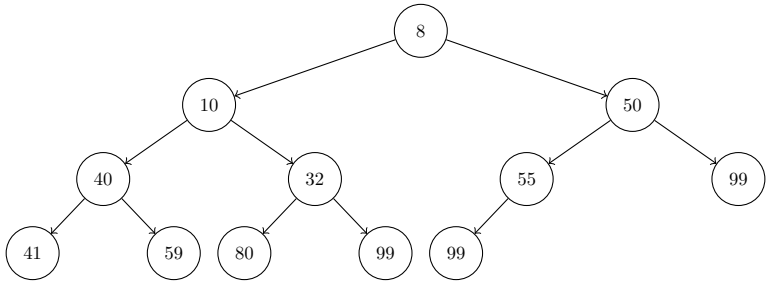
(a) I vilken ordning har elementen stoppats in? Det finns flera lösningar. Ge fyra korrekta lösningar för maxpoäng. (2)

(b) Var i tabellen skulle elementen ligga om i kvadratisk sondering (quadratic probing) skulle användas i stället för linjär? Utgå från en av de möjliga ordningarna du hittade i (a). (1)

(c) Om vi tar bort 11 från den delvis fyllda hashtabellen ovan (remove / delete), hur kommer tabellen se ut? Det finns flera olika lösningar. Ge två olika lösningar som inte hashar om hela tabellen utom möjligen i värsta fallet, och förklara dem, för maxpoäng. (2)

2. **Träd** (5 p)

Studera följande träd:



(a) Är trädet ett binärt sökträd? Motivera! (1)

(b) Är trädet komplett? Motivera! (1)

(c) Är trädet en max- eller min-heap? Motivera! (1)

(d) I vilken ordning besöks noderna om en levelorder-traversering av trädet görs? (2)

3. **Sorteringsalgoritmer** (5 p)

Svaren behöver ej motiveras.

Efter **ett fåtal** iterationer av några olika sorteringsalgoritmer på den osorterade arrayen *Original* i tabellen nedan (iteration i bemärkelse fullständig körning av inre loop, alternativt rekursivt anrop) har vi resultaten i *1, 2, 3, 4* och *5*.

Original:	82	29	52	99	34	62	6	94	46	61	43	71	65	91	58
1:	29	52	82	99	34	62	6	94	46	61	43	71	65	91	58
2:	6	29	34	82	43	52	99	46	62	58	94	61	65	71	91
3:	6	29	34	99	52	62	82	94	46	61	43	71	65	91	58
4:	82	61	71	52	43	65	58	29	46	34	6	62	91	94	99
5:	6	29	52	46	34	43	58	61	62	94	82	71	65	91	99
Sorterad:	6	29	34	43	46	52	58	61	62	65	71	82	91	94	99

Matcha delvis sorterad array mot en algoritm. Felaktig matchning ger minuspoäng, dock kan uppgiften ej ge total minuspoäng. Utelämnat svar ger ej minuspoäng. För full poäng krävs 5 korrekta svar.

(a) Selectionsort

(b) Insertionsort

(c) Heapsort

(d) Bubble sort, element bubblas från höger till vänster

(e) Quicksort, elementet längst till höger i partitionen används som pivot

4. **Algoritmer och tidskomplexitet** (5 p)

Visa kort hur du kommer fram till tidskomplexiteten på följande uppgifter.

(a) Beräkna tidskomplexiten med avseende på n för följande funktion: (1)

```
int f1(int n) {
  int result = 0;
  for (int i = 0; i < 2*n; i++) {
    result += i;
  }
  return result;
}
```

(b) Beräkna tidskomplexiten med avseende på n för följande funktion: (1)

```
int f2(int n) {
  int result = n;
  n = 20;
  for (int i = 0; i < n; i++) {
    result += i;
  }
  return result;
}
```

(c) Beräkna tidskomplexiten med avseende på n för följande funktion: (1)

```
int f3(int n) {
  int result = 0;
  for (int i = n; i > 0; i /= 2) {
    result += f1(n);
  }
  return result;
}
```

(d) Beräkna tidskomplexiten med avseende på n (`array.size()`) för `f4`: (2)

```
int sumRange(const vector<int> &array, int from, int to) {
  int result = 0;
  for (int i = from; i < to; i++) {
    result += array[i];
  }
  return result;
}

int f4(const vector<int> &array) {
  int result = 0;
  for (int i = 0; i < array.size() - 4; i++) {
    result += sumRange(array, i, i + 4);
  }
  return result;
}
```

5. **Kassa kassasystem** (10 p)

Matbutiken i den lilla staden Snålköping har nyligen beslutat sig för att ta steget in i 2000-talet genom att byta ut sitt gamla pappersbaserade system för att hantera köp och inventarier mot ett digitalt system. I och med att de vill lägga ut så lite pengar som möjligt har de noga undersökt marknaden och valt det billigaste systemet som uppfyllde deras krav. Tyvärr insåg inte ledningen i förhand att det var viktigt att kravställa prestandan hos systemet, så POS-systemet (Point of Sales) de köpte är otroligt långsamt på vissa operationer.

En typisk dag på matbutiken går till ungefär på följande sätt:

07:00-08:00 Varuleveransen för dagen kommer. Personalen matar in vilka nya varor som har köpts in i kassasystemet och ställer dem på rätt ställe i butiken.

08:00-18:00 Normala öppettider. Personalen matar in alla köp i kassan.

18:00-19:00 Efter stängning frågar personalen kassasystemet hur många exemplar av varje produkt som bör finnas i butiken, och jämför med antalet som faktiskt finns kvar för att se att allt har gått rätt till under dagen.

Problemet med det nya systemet är att det sista steget, att fråga systemet hur många av varje produkt som bör finnas kvar i butiken, tar väldigt lång tid för systemet att producera, ibland upp emot 30 minuter. Detta är såklart inte acceptabelt, eftersom personalen då måste stå och vänta innan de kan börja inventera (detta känner ledningen inte att de har råd med, de bor ju trots allt i Snålköping). Ledningen har fått nys om att du just har klarat DALG-tentan, och ber dig därmed att komma fram till en bättre lösning!

(a) Beskriv hur du kan designa ett kassasystem som inte är lika kasst som det som köptes in ovan. Butiken har ≈ 1000 olika typer av varor och ungefär mellan 5 och 200 exemplar av varje vara. Som beskrevs ovan behöver systemet snabbt hantera de tre fallen ovan. (4)

Beskriv hur systemet går till väga för att hantera dessa tre fall. Beskriv varje operation i punktform, där varje punkt består av en eller två meningar. Börja gärna med en kort (1-2 meningar) beskrivning av vilken indata som behövs. Avsluta vid behov med en kort (1-2 meningar) beskrivning av hur utdatan ska tolkas.

> **Exempel** (relaterat till uppgiften, men löser såklart inte uppgiften):
>
> Namnet på alla varor som kunden köper lagras i en array som vi kallar a.
>
> i. Initiera en heltalsvariabel, *sum*, till 0.
>
> ii. För varje element i a:
>
> A. Öka *sum* med 1.
>
> iii. Multiplicera *sum* med 2.
>
> *sum* innehåller nu antalet varor som kunden köpte gånger 2.

(b) Motivera kort (1-6 meningar) varför just dessa algoritmer är lämpliga i det här systemet. (1)

(c) Vilken eller vilka datastrukturer använder du för att lagra data om vilka varor som finns i butiken? Motivera kort (1-4 meningar) varför just dessa är lämpliga. (1)

(d) Vilken tidskomplexitet har din lösning för att beräkna vilka varor som bör finnas i butiken, uttryckt i antalet olika sorters varor som finns i butiken, n? Visa även hur du kommer fram till ditt svar. (1)

(e) Efter att ha använt ditt system ett tag kommer ledningen tillbaka till dig med ytterligare ett önskemål. För att effektivisera inventeringen skulle det vara bra om listan med antalet produkter som ska vara kvar i butiken var sorterad efter den ordning som (3)

produkterna står uppställda i butiken. För att ådstakomma detta har du fått tillgång till en lista med alla produkter som nu finns i butiken, sorterad så som ledningen vill att inventarielistan ska vara sorterad.

Beskriv, på samma form som i (a), hur du kan generera en inventarielista som är sorterad enligt den lista som ledningen gett dig.

6. **En utflykt i bergen** (10 p)

Efter mycket plugg under terminen har du beslutat dig att ta en välförtjänt semester i bergen under påskuppehållet. Ditt mål är att ta dig hela vägen upp till Sarek nationalpark. Du har beslutat dig för att ta tåget större delen av vägen, men sista biten har du beslutat att hyra en bil för att det ska gå smidigt. Det som oroar dig mest med resan är dock att du ska välja en dålig väg och inte ta dig fram hela vägen utan att behöva ladda din elbil på vägen (du har hört att det är mycket ont om laddningsställen på vägen). Du kan enkelt se att den totala sträckan du behöver köra är inom bilens räckvidd, det som oroar dig är det faktum att det ibland är mycket brant terräng, och att bilen drar mer när den kör i uppförsbacke än när den kör på plan mark.

För att försöka komma fram till om det är möjligt att ta sig fram, samt hur du i så fall vill köra har du sammanställt en tabell med alla möjliga vägar i området och deras lutning. Nedan följer en del av tabellen:

Från	**Till**	**Avstånd**	**Höjdskillnad**
Luleå	Boden	36 km	100 m
Luleå	Överkalix	105 km	210 m
Boden	Arjeplog	225 km	100 m
Överkalix	Gällivare	142 km	400 m
...	...	...	...

Första raden säger exempelvis att det går att åka från Luleå till Boden. Den vägen är 36 km, och Boden ligger 100 m över Luleå. Självklart går det att åka åt båda hållen på vägarna, men åker man åt andra hållet åker man i stället nedåt, vilket inte kostar någon extra energi.

Du har frågat tillverkaren av bilen angående förbrukningen i uppförsbacke, och du har fått följande approximation: $e = 0.1s + 0.5h$ där s är hur långt man har åkt (i kilometer), h är höjdskillnaden från start till mål, och e är hur mycket energi som går åt (i amperetimmar, Ah). Detta gäller endast i uppförsbacke. I nedförsbacke (dvs. om man åker åt andra hållet mot vad som står i tabellen) används i stället $e = 0.1s$ (dvs. $h = 0$) eftersom bilen inte är tillräckligt modern för att kunna återanvända den extra energin från nedfarten.

Med denna kunskapen vill du skriva ett program som kan hitta en väg från en startpunkt till en slutpunkt som gör att bilen förbrukar så lite energi som möjligt (du vill ha så mycket marginal som möjligt ifall något skulle hända). Med hjälp av det kan du sedan se om batteriet med en kapacitet av 1000 Ah räcker, eller om du behöver hyra en annan bil.

Tips: Om du tycker att enheten amperetimmar (Ah) är abstrakt kan du tänka dig liter bensin i stället.

(a) Beskriv hur du effektivt kan hitta den väg som förbrukar så lite energi som möjligt. Antag att du har en tabell med ≈ 5000 olika vägsegment i. (4)

Beskriv algoritmen i punktform, där varje punkt består av en eller två meningar. Börja gärna med en kort (1-2 meningar) beskrivning av vilken indata som behövs. Avsluta gärna med en kort (1-2 meningar) beskrivning av hur hur utdatan kan användas för att lösa problemet.

Exempel (relaterat till uppgiften, men löser såklart inte uppgiften):

Tabellen från uppgiften lagras i en array som vi kallar a.

i. Plocka ut det första elementet ur a, lagra det i b.

ii. Öka avståndet i b med 1 km.

b innehåller nu ett avståndet till någon stad, med 1 km marginal.

(b) Motivera kort (1-4 meningar) varför just denna algoritm är lämplig för att lösa det här problemet. (1)

(c) Vilka datastrukturer använder du i din lösning? Motivera kort (1-4 meningar) varför just dessa är lämpliga i din lösning. (1)

(d) Vilken tidskomplexitet har din lösning uttryckt i det totala antalet rader i tabellen n och antalet städer som finns i tabellen m? Visa även hur du kommer fram till ditt svar. (1)

(e) Efter att ha pratat om dina problem med en kompis har du kommit över en lista med städer som erbjuder laddningsmöjligheter. Beskriv, på samma form som i (a), hur du kan hitta en väg som garanterat tar dig från punkt A till punkt B givet att din bil har en kapacitet på c Ah, och att du kan ladda bilen full vid någon av städerna i listan din kompis gav dig. (3)